



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/637,169	08/08/2003	Marc Tremblay	SUN-P9325-MEG	2951
57960	7590	09/07/2006	EXAMINER	
SUN MICROSYSTEMS INC. C/O PARK, VAUGHAN & FLEMING LLP 2820 FIFTH STREET DAVIS, CA 95618-7759			FENNEMA, ROBERT E	
			ART UNIT	PAPER NUMBER
			2183	

DATE MAILED: 09/07/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

10/637,169

Applicant(s)

TREMBLAY ET AL.

Examiner

Robert E. Fennema

Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 20 July 2006.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-4, 6-16 and 18-25 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-4, 6-16, and 18-25 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|---|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____ |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| Paper No(s)/Mail Date. _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. Claims 1-4, 6-16, and 18-25 have been considered. Claims 1, 13, and 25 have been amended as per Applicants request.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1-4, 6-16, and 18-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Moss et al ("Transactional Memory: Architectural Support for Lock-Free Data Structures", herein Moss), in view of Oplinger et al ("Enhancing Software Reliability with Speculative Threads", herein Oplinger).

4. As per Claim 1, Moss teaches: A method for executing a fail instruction to facilitate transactional execution on a processor, comprising:

wherein changes made during the transactional execution are not committed to the architectural state of the processor unless the transactional execution successfully completes (Section 2.1, it requires that the commit instruction be successful); and

if the fail instruction is encountered during the transactional execution, terminating the transactional execution without committing results of the transactional

execution to the architectural state of the processor (Section 2.1, see commit, abort and validate instructions), but fails to teach:

executing a start transactional execution instruction to start transactionally
executing a block of instructions within a program;

wherein terminating the transactional execution involves branching to a location specified by the fail instruction.

However, Oplinger teaches a transactional programming model in which his abort/fail instruction not only eliminates the speculative state, but also branches the program to an address that was previously set by a TRY instruction (Section 3.2). The advantage of having an instruction is being able to go to an error handling case in the event of an abort (see Section 3.2, the second code example, where the system jumps to an error message on an abort), giving the programmer more control over the execution and troubleshooting of his program. Given these advantages, it would have been obvious to one of ordinary skill in the art at the time the invention was made to take Moss's invention, and allow the abort instruction to specify an address to branch to in the event of the abort instruction executing.

5. As per Claim 2, Moss teaches: The method of claim 1, wherein terminating the transactional execution involves discarding changes made during the transactional execution (Section 2.1).

6. As per Claim 3, Moss teaches: The method of claim 2, wherein discarding changes made during the transactional execution involves:

discarding register file changes made during the transactional execution (Section 2.1, see commit, abort, and validate instructions);

clearing load marks from cache lines (Section 3.1.2, XABORT tagged entries are set to EMPTY);

draining store buffer entries generated during transactional execution (Section 3.1.2, XABORT tagged entries are set to EMPTY, clearing out the data that was temporarily stored in them); and

clearing store marks from cache lines (Section 3.1.2, XABORT tagged entries are set to EMPTY).

7. As per Claim 4, Oplinger teaches the method of claim 1 wherein terminating the transactional execution additionally involves branching to a location specified by a corresponding start transactional execution (STE) instruction (Section 3.2).

8. As per Claim 6, Moss teaches: The method of claim 1, wherein terminating the transactional execution additionally involves attempting to re-execute the block of instructions (Section 2.2, wherein if Step 4 fails, the process repeats at step 1).

9. As per Claim 7, Moss teaches: The method of claim 1, wherein if the transactional execution of the block of instructions is successfully completes, the method further comprises:

atomically committing changes made during the transactional execution (Sections 2.0 and 2.1); and

resuming normal non-transactional execution (As Section 2.2 says, the transactional execution is intended for critical sections, which are small parts of non-transactional code blocks. Therefore, when it was finished, it would resume execution in the non-transactional code. Section 5.4 further elaborates on this, by stating that the transactions have short durations, and small data sets, meaning that it must go to non-transactional after that short duration).

10. As per Claim 8, Moss teaches: The method of claim 1, wherein potentially interfering data accesses from other processes are allowed to proceed during the transactional execution of the block of instructions (Section 1, where it is stated that "If one process is interrupted in the middle of an operation, other processes will not be prevented from operating on that object").

11. As per Claim 9, Moss teaches: The method of claim 1, wherein if an interfering data access from another process is encountered during the transactional execution, the method further comprises:

discarding changes made during the transactional execution (Section 2.1, see commit, abort, and validate instructions); and
attempting to re-execute the block of instructions (Section 2.2, see step 4).

12. As per Claim 10, Moss teaches: The method of claim 1, wherein the block of instructions to be executed transactionally comprises a critical section (Section 2.2, first paragraph).

13. As per Claim 11, Moss teaches: The method of claim 1, wherein the fail instruction is a native machine code instruction of the processor (Section 7).

14. As per Claim 12, Moss teaches: The method of claim 1, wherein the fail instruction is defined in a platform-independent programming language (Section 3.1 and 3.2, which show an example written in C).

15. As per Claim 13, Moss teaches: A computer system that supports a fail instruction to facilitate transactional execution, comprising:
a processor (inherent in a computer system that executes instructions); and
an execution mechanism within the processor (inherent in a computer system that executes instructions);

wherein changes made during the transactional execution are not committed to the architectural state of the processor unless the transactional execution successfully completes (Section 2.1, it requires that the commit instruction be successful); and

wherein if the fail instruction is encountered during the transactional execution, the execution mechanism is configured to terminate the transactional execution without committing results of the transactional execution to the architectural state of the processor (Section 2.1, see commit, abort and validate instructions), but fails to teach:

wherein the execution mechanism is configured to execute a start transactional execution instruction to transactionally execute a block of instructions within a program;

wherein terminating the transactional execution involves branching to a location specified by the fail instruction.

However, Oplinger teaches a transactional programming model in which his abort/fail instruction not only eliminates the speculative state, but also branches the program to an address that was previously set by a TRY instruction (Section 3.2). The advantage of having an instruction is being able to go to an error handling case in the event of an abort (see Section 3.2, the second code example, where the system jumps to an error message on an abort), giving the programmer more control over the execution and troubleshooting of his program. Given these advantages, it would have been obvious to one of ordinary skill in the art at the time the invention was made to take Moss's invention, and allow the abort instruction to specify an address to branch to in the event of the abort instruction executing.

16. As per Claim 14, Moss teaches: The computer system of claim 13, wherein while terminating the transactional execution, the execution mechanism is configured to discard changes made during the transactional execution (Section 2.1).

17. As per Claim 15, Moss teaches: The computer system of claim 14, wherein while discarding changes made during the transactional execution, the execution mechanism is configured to:

discard register file changes made during the transactional execution (Section 2.1);

clear load marks from cache lines (Section 3.1.2, XABORT tagged entries are set to EMPTY);

drain store buffer entries generated during transactional execution (Section 3.1.2, XABORT tagged entries are set to EMPTY, clearing out the data that was temporarily stored in them); and

18. to clear store marks from cache lines (Section 3.1.2, XABORT tagged entries are set to EMPTY).

19. As per Claim 16, Oplinger teaches the computer system of claim 13, but fails to teach: wherein while terminating the transactional execution, the execution mechanism is additionally configured to branch to a location specified by a corresponding start transactional execution (STE) instruction (Section 3.2).

20. As per Claim 18, Moss teaches: The computer system of claim 13, wherein while terminating the transactional execution, the execution mechanism is additionally configured to attempt to re-execute the block of instructions (Section 2.2, wherein if Step 4 fails, the process repeats at step 1).

21. As per Claim 19, Moss teaches: The computer system of claim 13, wherein if the transactional execution of the block of instructions is successfully completes, the execution mechanism is configured to:

atomically commit changes made during the transactional execution (Sections 2.0 and 2.1); and

to resume normal non-transactional execution (As Section 2.2 says, the transactional execution is intended for critical sections, which are small parts of non-transactional code blocks. Therefore, when it was finished, it would resume execution in the non-transactional code. Section 5.4 further elaborates on this, by stating that the transactions have short durations, and small data sets, meaning that it must go to non-transactional after that short duration).

22. As per Claim 20, Moss teaches: The computer system of claim 13, wherein the computer system is configured to allow potentially interfering data accesses from other processes to proceed during the transactional execution of the block of instructions (Section 1, where it is stated that "If one process is interrupted in the middle of an

operation, other processes will not be prevented from operating on that object”).

23. As per Claim 21, Moss teaches: The computer system of claim 13, wherein if an interfering data access from another process is encountered during the transactional execution, the execution mechanism is configured to:

discard changes made during the transactional execution (Section 2.1, see commit, abort, and validate instructions); and

to attempt to re-execute the block of instructions (Section 2.2, see step 4).

24. As per Claim 22, Moss teaches: The computer system of claim 13, wherein the block of instructions to be executed transactionally comprises a critical section (Section 2.2, first paragraph).

25. As per Claim 23, Moss teaches: The computer system of claim 13, wherein the fail instruction is a native machine code instruction of the processor (Section 7).

26. As per Claim 24, Moss teaches: The computer system of claim 13, wherein the fail instruction is defined in a platform-independent programming language (Section 3.1 and 3.2, which show an example written in C).

27. As per Claim 25, Moss teaches: A computing means that supports that supports a fail instruction to facilitate transactional execution, comprising:

a processing means (inherent in a computer that executes instructions); and
an execution means within the processing means (inherent in a computer that executes instructions);

wherein changes made during the transactional execution are not committed to the architectural state of the processor unless the transactional execution successfully completes (Section 2.1, it requires the commit instruction to successfully complete); and

wherein if the fail instruction is encountered during the transactional execution, the execution means is configured to terminate the transactional execution without committing results of the transactional execution to the architectural state of the processor (Section 2.1, see the commit, abort, and validate instructions), but fails to teach:

wherein the execution means is configured to execute a start transactional execution instruction to transactionally execute a block of instructions within a program;

wherein terminating the transactional execution involves branching to a location specified by the fail instruction.

However, Oplinger teaches a transactional programming model in which his abort/fail instruction not only eliminates the speculative state, but also branches the program to an address that was previously set by a TRY instruction (Section 3.2). The advantage of having an instruction is being able to go to an error handling case in the event of an abort (see Section 3.2, the second code example, where the system jumps to an error message on an abort), giving the programmer more control over the execution and troubleshooting of his program. Given these advantages, it would have

been obvious to one of ordinary skill in the art at the time the invention was made to take Moss's invention, and allow the abort instruction to specify an address to branch to in the event of the abort instruction executing.

Response to Arguments

28. Applicant's arguments filed 7/20/2006 have been fully considered but they are not persuasive. Applicant has argued that Oplinger teaches a software system using try-catch blocks to implement transaction execution, while the present invention uses a hardware-implemented start transactional execution instruction to start transactional execution of a section of code. Examiner disagrees, and points to Oplinger's section 3.2, which describes the TRY instruction. The TRY instruction is a machine instruction, which "indicates the start of a transaction". Not only does the TRY instruction indicate that the section of code should begin transactional execution, but it is also a machine (hardware-implemented) instruction. Given this disclosure of the TRY statement used by Oplinger, the prior art continues to read on the claims in question. The amended claim rejections have been amended to account for Oplinger's TRY instruction indicating the start of the transactional section, as opposed to Moss's section 2.1.

Conclusion

29. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP


Art Unit: 2183

§ 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the date of this final action.

Robert E Fennema
Examiner
Art Unit 2183

RF



EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100